TopFormer: Token Pyramid Transformer for Mobile Semantic Segmentation

Wenqiang Zhang[*] Zilong Huang[†] Guozhong Luo[†] Tao Chen[‡]
Xinggang Wang[*] Wenyu Liu[*] Gang Yu[†] Chunhua Shen[§]

Abstract

*Although vision transformers (ViTs) have achieved great success in computer vision, the heavy computational cost hampers their applications to dense prediction tasks such as semantic segmentation on mobile devices. In this paper, we present a mobile-friendly architecture named **To**ken **P**yramid Vision Trans**former** (**TopFormer**). The proposed **TopFormer** takes Tokens from various scales as input to produce scale-aware semantic features, which are then injected into the corresponding tokens to augment the representation. Experimental results demonstrate that our method significantly outperforms CNN- and ViT-based networks across several semantic segmentation datasets and achieves a good trade-off between accuracy and latency. On the ADE20K dataset, TopFormer achieves 5% higher accuracy in mIoU than MobileNetV3 with lower latency on an ARM-based mobile device. Furthermore, the tiny version of TopFormer achieves real-time inference on an ARM-based mobile device with competitive results. The code and models are available at:*

https://github.com/hustvl/TopFormer.

1. Introduction

Vision Transformers (ViTs) have shown considerably stronger results for a few vision tasks, such as image classification [11], object detection [28], and semantic segmentation [58]. Despite the success, the Transformer architecture with the full-attention mechanism [42] requires powerful computational resources beyond the capabilities of many mobile and embedded devices. In this paper, we aim to explore a mobile-friendly Vision Transformer specially designed for dense prediction tasks, *e.g.*, semantic segmentation.

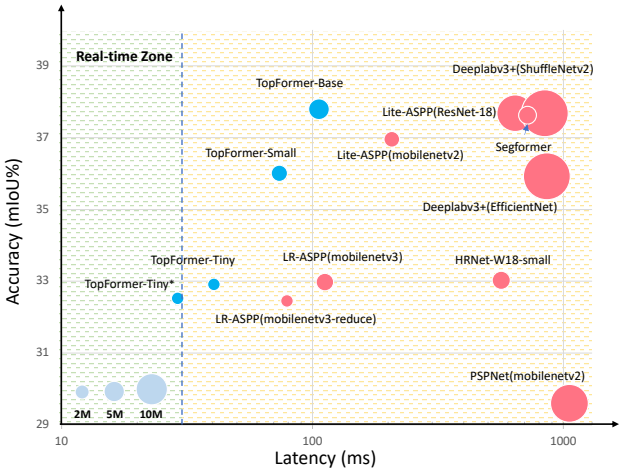

Figure 1 – The latency, mIoU performance versus model size on the ADE20K val. set. Previous models are marked as red points, and our models are shown in blue points. Our methods achieve a better latency/accuracy trade-off. The latency is measured on a single Qualcomm Snapdragon 865 with input size 512×512, and only an ARM CPU core is used for speed testing. No other means of acceleration, *e.g.*, GPU or quantification, is used. * indicates the input size is 448×448.

To adapt Vision Transformers to various dense prediction tasks, most recent Vision Transformers such as PVT [43], CvT [45], LeViT [12], MobileViT [31] adopt a hierarchical architecture, which is generally used in Convolution Neural Networks (CNNs), *e.g.*, AlexNet [23], ResNet [15]. These Vision Transformers apply the global self-attention and its variants on the high-resolution tokens, which bring heavy computation cost due to the quadratic complexity in the number of tokens.

To improve the efficiency, some recent works, *e.g.*, Swin Transformer [28], Shuffle Transformer [19], Twins [7] and HR-Former [51], compute self-attention within the local/windowed region. However, the window partition is surprisingly time-consuming on mobile devices. Besides, Token slimming [40] and Mobile-Former [6] decrease calculation capacity by reducing the number of tokens, but sacrifice their recognition accuracy.

Among these Vision Transformers, MobileViT [31] and Mobile-Former [6] are specially designed for mobile de-

[*]Huazhong University of Science and Technology, China
[†]Tencent PCG, China
[‡]Fudan University, China
[§]Zhejiang University, China
W. Zhang and Z. Huang contributed equally. This work was done when W. Zhang was an intern at Tencent PCG. X. Wang and W. Liu are the corresponding authors, e-mail: {xgwang,liuwy}@hust.edu.cn

vices. They both combine the strengths of CNNs and ViTs. For image classification, MobileViT achieves better performance than MobileNets [16, 36] with a similar number of parameters. Mobile-Former achieves better performance than MobileNets with a fewer number of FLOPs. However, they do not show advantages in actual latency on mobile devices compared to MobileNets, as reported in [31]. It raises a question: *Is it possible to design mobile-friendly networks which could achieve better performance on mobile semantic segmentation tasks than MobileNets with lower latency?*

Inspired by MobileViT and Mobile-Former, we also make use of the advantages of CNNs and ViTs. A CNN-based module, denoted as Token Pyramid Module, is used to process high-resolution images to produce local features[1] pyramid quickly. Considering the very limited computing power on mobile devices, here we use a few stacked light-weight MobileNetV2 blocks with a fast down-sampling strategy to build a token pyramid. To obtain rich semantics and large receptive field, the ViT-based module, denoted as Semantics Extractor, is adopted and takes the tokens as input. To further reduce the computational cost, the average pooling operator is used to reduce tokens to an extremely small number, *e.g.*, $1/(64\times64)$ of the input size. Different from ViT [11], T2T-ViT [49] and LeViT [12] use the last output of the embedding layer as input tokens, we pool the tokens from different scales (stages) into the very small numbers (resolution) and concatenate them along the channel dimension. Then the new tokens are fed into the Transformer blocks to produce global semantics. Due to the residual connections in the Transformer block, the learned semantics are related to scales of tokens, denoted as scale-aware global semantics.

To obtain powerful hierarchical features for dense prediction tasks, scale-aware global semantics is split by the channels of tokens from different scales, then the scale-aware global semantics are fused with the corresponding tokens to augment the representation. The augmented tokens are used as the input of the segmentation head.

To demonstrate the effectiveness of our approach, we conduct experiments on the challenging segmentation datasets: ADE20K [59], Pascal Context [33] and COCO-Stuff [1]. We examine the latency on hardware, *i.e.*, an off-the-shelf ARM-based computing core. As shown in Figure 1, our approach obtains better results than MobileNets with lower latency. To demonstrate the generalization of our approach, we also conduct experiments of object detection on the COCO [27] dataset. To summarize, our contributions are as follows.

- The proposed TopFormer takes tokens from different scales as input, and pools the tokens to the very small numbers, in order to obtain scale-aware semantics with very light computation cost.
- The proposed Semantics Injection Module can inject the scale-aware semantics into the corresponding tokens to build powerful hierarchical features, which is critical to dense prediction tasks.
- The proposed base model can achieve 5% mIoU better than that of MobileNetV3, with lower latency on an ARM-based mobile device on the ADE20K dataset. The tiny version can perform real-time segmentation on an ARM-based mobile device, with competitive results.

[1]We use 'features' and 'tokens' interchangeably here.

2. Related Work

In this section, we review recent approaches in terms of three aspects: 1) Light-weight Vision Transformer, 2) Efficient Convolutional Neural Networks, 3) Mobile Semantic Segmentation.

2.1. Light-weight Vision Transformers

There are many explorations [18, 44, 55] for the use of transformer structures in image recognition. ViT [11] is the first work to apply a pure transformer to image classification, achieving state-of-the-art performance. Following that, DeiT [41] introduces token-based distillation to reduce the amount of data necessary for training the transformer. T2T-ViT [49] structures the image to tokens by recursively aggregating neighboring tokens into one token to reduce tokens length. Swin Transformer [28] computes self-attention within each local window, resulting in linear computational complexity in the number of input tokens. However, these Vision Transformers and the follow-ups are often of a large number of parameters and heavy computation complexity.

To build a light-weight Vision Transformer, LeViT [12] designs a hybrid architecture that uses stacked standard convolution layers with stride-2 to reduce the number of tokens, then appends an improved Vision Transformer to extract semantics. For classification task, LeViT can significantly outperform EfficientNet on CPU. MobileViT [31] adopts the same strategy and uses the MobilenetV2 block instead of the standard convolution layer for downsampling the feature maps. Mobile-Former takes parallel structure with a bidirectional bridge and leverages the advantage of both MobileNet and transformer. However, the MobileViT and other ViT-based networks are significantly slower than MobileNets [16, 17, 36] on mobile devices, as reported in [31]. For the segmentation task, the input images are always of high-resolution. Thus it is even more challenging for ViT-based networks to execute faster than MobileNets. In this paper, we aim to design a light-weight Vision Transformer which can outperform MobileNets with lower latency for the segmentation task.

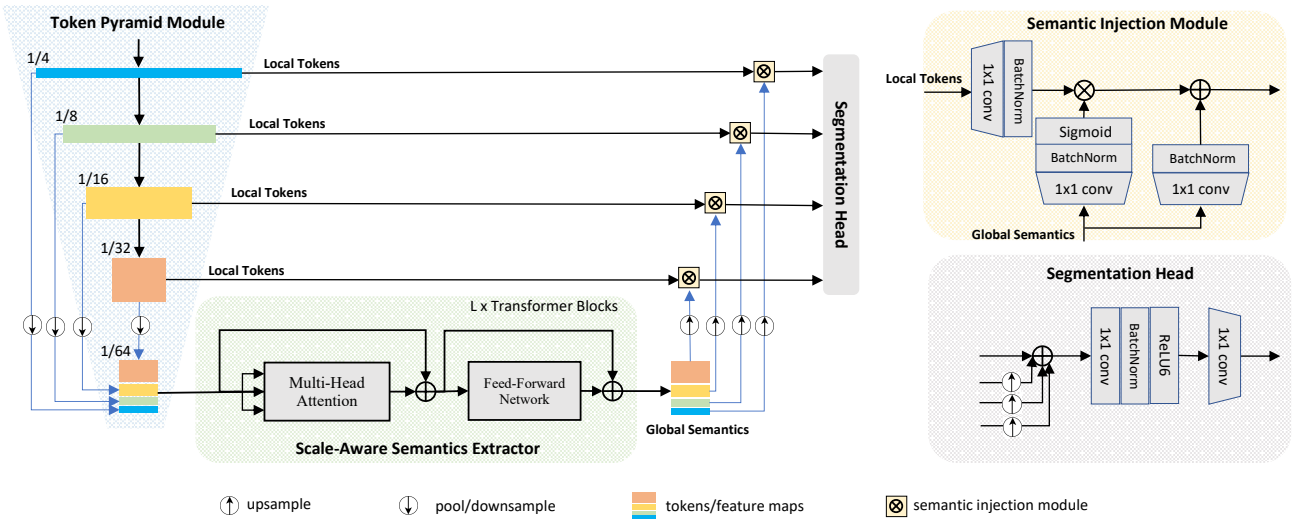

Figure 2 – The architecture of the proposed Token Pyramid Transformer.

2.2. Efficient Convolutional Neural Networks

The increasing need of deploying vision models on mobile and embedded devices encourages the study on efficient Convolutional Neural Networks designs. MobileNet [16,17,36] proposes an inverted bottleneck structure which stacks depth-wise and point-wise convolutions. IGCNet [53] and ShuffleNet [30,54] use channel shuffle/permutation operators to the channel to make cross-group information flow for multiple group convolution layers. GhostNet [13] uses the cheaper operator, depth-wise convolutions, to generate more features. AdderNet [2] utilizes additions to trade massive multiplications. MobileNeXt [60] flips the structure of the inverted residual block and presents a building block that connects high-dimensional representations instead. EfficientNet [38, 39] and TinyNet [14] study the compound scaling of depth, width and resolution.

2.3. Mobile Semantic Segmentation

The most accurate segmentation networks usually require computation at billions of FLOPs, which may exceed the computation capacity of the mobile and embedded devices. To speed up the segmentation and reduce the computational cost, ICNet [56] uses multi-scale images as input and a cascade network to be more efficient. DFANet [24] utilizes a light-weight backbone to speed up its network and proposes a cross-level feature aggregation to boost accuracy. SwiftNet [35] uses lateral connections as the cost-effective solution to restore the prediction resolution while maintaining the speed. BiSeNet [47] introduces the spatial path and the semantic path to reduce computation. AlignSeg [20] and SFNet [25] align feature maps from adjacent levels and further enhances the feature maps using a feature pyramid framework. ESPNets [32] save computation by decomposing standard convolution into point-wise convolution and spatial pyramid of dilated convolutions. AutoML techniques [5, 10, 34] are used to search for efficient architectures for scene parsing. NRD [52] dynamically generates the neural representations with dynamic convolution filter networks. LRASPP [16] adopts MobileNetV3 as encoder and proposes a new efficient segmentation decoder Lite Reduced Atrous Spatial Pyramid Pooling (LR-ASPP), and it still serves as a strong baseline for mobile semantic segmentation.

3. Architecture

Our overall network architecture is shown in Figure 2. As we can see, our network consists of several parts: Token Pyramid Module, Semantics Extractor, Semantics Injection Module and Segmentation Head. The Token Pyramid Module takes an image as input and produces the token pyramid. The Vision Transformer is used as a semantics extractor, which takes the token pyramid as input and produces scale-aware semantics. The semantics are injected into the tokens of the corresponding scale for augmenting the representation by the Semantics Injection Module. Finally, Segmentation Head uses the augmented token pyramid to perform the segmentation task. Next, we present the details of these modules.

3.1. Token Pyramid Module

Inspired by MobileNets [36], the proposed Token Pyramid Module consists of stacked MobileNet blocks [36]. Different from MobileNets, the Token Pyramid Module does not aim to obtain rich semantics and large receptive field, but uses fewer blocks to build a token pyramid. We show the layer settings of the Token Pyramid Module in Subsection 3.4.

As shown in Figure 2, taking an image $\mathbf{I} \in \mathbb{R}^{3\times H\times W}$ as

input, where $3, H, W$ indicate the RGB channels, height, width of $\mathbf{I}$ respectively, our Token Pyramid Module first passes the image through some MobileNetV2 blocks [36] to produce a series of tokens $\{\mathbf{T}^1, ..., \mathbf{T}^N\}$, where N indicates the number of scales[2]. Afterwards, the tokens $\{\mathbf{T}^1, ..., \mathbf{T}^N\}$, are average pooled to the target size, *e.g.*, $\mathbb{R}^{\frac{H}{64} \times \frac{W}{64}}$. Finally, the tokens from different scales are concatenated along the channel dimension to produce the new tokens. The new tokens will be fed into the Vision Transformer to produce Scale-aware Semantics. Because the new tokens are of small quantity, the Vision Transformer can run with very low computation cost even if the new tokens have large channels.

3.2. Vision Transformer as Scale-aware Semantics Extractor

The Scale-aware Semantics Extractor consists of a few stacked Transformer blocks. The number of Transformer blocks is L. The Transformer block consists of the multi-head Attention module, the Feed-Forward Network (FFN) and residual connections. To keep the spatial shape of tokens and reduce the numbers of reshape, we replace the Linear layers with a 1×1 convolution layer. Besides, all of TopFormer's non-linear activations are ReLU6 [17] instead of GELU function in ViT.

For the Multi-head Attention module, we follow the settings of LeViT [12], and set the head dimension of keys K and queries Q to have $D = 16$, the head of values V to have $2D = 32$ channels. Decreasing the channels of K and Q will reduce computational cost when calculating attention maps and output. Meanwhile, we also drop the Layer normalization layer and append a batch normalization to each convolution. The batch normalization can be merged with the preceding convolution during inference, which can run faster over layer normalization.

For the Feed-Forward Network, we follow [19, 48] to enhance the local connections of Vision Transformer by inserting a depth-wise convolution layer between the two 1×1 convolution layers. The expansion factor of FFN is set to 2 to reduce the computational cost. The number of Transformer blocks is L and then the number of heads will be given in subsection 3.4.

As shown in Figure 2, the Vision Transformer takes the tokens from different scales as input. To further reduce the computation, the average pooling operator is used to reduce the numbers of tokens from different scales to $\frac{1}{64 \times 64}$ of the input size. The pooled tokens from different scales have the same resolution, and they are concatenated together as the input of the Vision Transformer. The Vision Transformer can obtain full-image receptive field and rich semantics. To be more specific, the global self-attention exchanges information among tokens along the spatial dimension. The 1×1 convolution layer will exchange information among tokens from different scales. In each Transformer block, the residual mapping is learned after exchanging information of tokens from all scales, then residual mapping is added into tokens to augment the representation and semantics. Finally, the scale-aware semantics are obtained after passing through several transformer blocks.

[2]We say that the tokens from different stages belong to different scales.

3.3. Semantics Injection Module and Segmentation Head

After obtaining the scale-aware semantics, we add them with the other tokens T^N directly. However, there is a significant semantic gap between the tokens $\{\mathbf{T}^1, ..., \mathbf{T}^N\}$ and the scale-aware semantics. To this end, the Semantics Injection Module is introduced to alleviate the semantic gap before fusing these tokens. As shown in Fig. 2, the Semantics Injection Module (SIM) takes the local tokens of the Token Pyramid module and the global semantics of the Vision Transformer as input. The local tokens are passed through the 1×1 convolution layer, followed by a batch normalization to produce the feature to be injected. The global semantics are fed into the 1×1 convolution layer followed by a batch normalization layer and a sigmoid layer to produce semantics weights, meanwhile, the global semantics also passed through the 1×1 convolution layer followed by a batch normalization. The three outputs have the same size. Then, the global semantics are injected into the local tokens by Hadamard production and the global semantics are also added with the feature after the injection. The outputs of the several SIMs share the same number of channels, denoted as M.

After the semantics injection, the augmented tokens from different scales capture both rich spatial and semantic information, which is critical for semantic segmentation. Besides, the semantics injection alleviates the semantic gap among tokens. The proposed Segmentation head firstly upsamples the low-resolution tokens to the same size as the high-resolution tokens and element-wise sums up the tokens from all scales. Finally, the feature is passed through two convolutional layers to produce the final segmentation map.

3.4. Architecture and Variants

To customize the network of various complexities, we introduce TopFormer-Tiny (TopFormer-T), TopFormer-Small (TopFormer-S) and TopFormer-Base (TopFormer-B), respectively.

The model size and FLOPs of the base, small and tiny models are given in the Table 1. The base, small and tiny models have 8, 6 and 4 heads in each multi-head self-attention module, respectively, and have $M = 256$, $M = 192$ and $M = 128$ as the target numbers of channels. For more details of network configures, please refer to

Method	Encoder	Params(M)	FLOPs(G)	mIoU	Latency(ms)
FCN-8s [29]	MobileNetV2	9.8	39.6	19.7	1015
PSPNet [57]	MobileNetV2	13.7	52.2	29.6	1065
DeepLabV3+ [4]	MobileNetV2	15.4	25.8	38.1	1035
DeepLabV3+ [4]	EfficientNet	17.1	26.9	36.2	970
DeepLabV3+ [4]	ShuffleNetV2-1.5x	16.9	15.3	37.6	960
Lite-ASPP [4]	ResNet18	12.5	19.2	37.5	648
Lite-ASPP [4]	MobileNetV2	2.9	4.4	36.6	235
R-ASPP [36]	MobileNetV2	2.2	2.8	32.0	177
LR-ASPP [16]	MobileNetV3-Large	3.2	2.0	33.1	126
LR-ASPP [16]	MobileNetV3-Large-reduce	1.6	1.3	32.3	81
HRNet-W18-Small [50]	HRNet-W18-Small	4.0	10.2	33.4	639
HR-NAS-A [10]	Searched	2.5	1.4	33.2	-
HR-NAS-B [10]	Searched	3.9	2.2	34.9	-
Segformer [46]	MiT-B0	3.8	8.4	37.4	770
Semantic FPN [22]	ConvMLP-S	12.8	33.8	35.8	777
Ours	TopFormer-T†	1.4	0.5	32.5	32
Ours	TopFormer-T	1.4	0.6	32.8	43
Ours	TopFormer-S	3.1	1.2	36.1	74
Ours	TopFormer-B	5.1	1.8	37.8	110

Table 1 – Results on ADE20K *val* set. Latency and GFLOPs calculation adopt images with 512×512 resolution as input. † indicates results are obtained with 448×448 resolution as input. Latency is measured based on a single Qualcomm Snapdragon 865 processor. All results are evaluated with single thread. Following the settings in MMSegmentation, batch size=32 is used for the CNN-based methods. For a fair comparison with Segformer, the batch size=16 is used for TopFormer. The mIoU is reported with single-scale inference.

the supplementary materials.

To achieve better trade-offs between accuracy and the actual latency, we choose the tokens from last three scales T^2, T^3 and T^4 as the inputs of SIM and the segmentation head.

4. Experiments

In this section, we first conduct experiments on several public datasets. We describe implementation details and compare results with other works for semantic segmentation tasks. We then conduct ablation studies to analyze the effectiveness and efficiency of different parts. Finally, we report the performance on object detection task to show the generalization ability of our method.

4.1. Semantic Segmentation

4.1.1 Datasets

We perform experiments over three datasets, ADE20K [59], PASCAL Context [33] and COCO-Stuff [1]. The mean of class-wise intersection over union (mIoU) is set as our evaluation metric. The full-precision TopFormer models are converted to TNN [9], the latency is then measured on an ARM-based computing core. **ADE20K**: The ADE20K dataset contains 25K images in total, covering 150 categories. All images are split into 20K/2K/3K for training, validation, and testing. **PASCAL Context**: The Pascal Context dataset has 4998 scene images for training and 5105 images for testing. There are 59 semantic labels and 1 background label. **COCO-Stuff**: The COCO-Stuff [1] dataset augments COCO dataset with pixel-level stuff annotations. There are 10000 complex images selected from COCO. The training set and test set consist of 9K and 1K images respectively.

4.1.2 Implementation Details

Our implementation is built upon MMSegmentation [8] and Pytorch. It utilizes ImageNet pre-trained TopFormer[3] as the backbone. The standard BatchNorm [21] layer is replaced by the Synchronize BatchNorm to collect the mean and standard-deviation of BatchNorm across multiple GPUs during training. For ADE20K dataset, we follow Segformer to use 160K scheduler and batch size is 16. The training iteration of COCO-Stuff and PASCAL Context is 80K. For all methods and datasets, the initial learning rate is set as 0.00012 and weight decay is 0.01. A "poly" learning rate scheduled with factor 1.0 is used. On ADE20K, we adopt the same data augmentation strategy as [46] for fair comparison. The training images are augmented by first randomly scaling and then randomly cropping out the fixed size patches from the resulting images. In addition, we also apply random resize, random horizontal flipping, random cropping etc. On COCO-Stuff and PASCAL Context, we use the default augmentation strategy of [8]. We resize

[3]The TopFormer-base achieve 75.3% Top-1 acc. on ImageNet-1K.

Input	Top-1 acc.	mIoU	params(M)
Token Pyramid	75.3	37.8	5.1
Token of the last scale	74.8	36.6	5.2

Table 2 – Ablation studies on Token Pyramid as input.

Output	mIoU	params(M)	FLOPs(G)
$\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}\}$	38.3	5.1	3.32
$\{\frac{1}{8}, \frac{1}{16}, \frac{1}{32}\}$	37.8	5.1	1.82
$\{\frac{1}{16}, \frac{1}{32}\}$	36.4	5.0	1.41

Table 3 – Ablation studies on Token Pyramid as output.

and crop the images to 480×480 for PASCAL Context and 512×512 for COCO-Stuff during training. Finally, we report the single scale results on validation set to compare with other methods. During inference, we follow the common strategy to rescale the short side of images to training cropping size for ADE20K and COCO-Stuff. As for PASCAL Context, the images are resized to 480×480 and then fed into our network.

4.1.3 Experiments on ADE20K

We compare our TopFormer with the previous approaches on the ADE20K validation set in Table 1. Actual latency is measured on the mobile device with a single Qualcomm Snapdragon 865 processor. Here, we choose light-weight vision transformers (ViT) [10, 22, 30, 46] and efficient convolution neural networks (CNNs) [15, 16, 36, 37] as the encoder. Besides, the various decoders are also included in Table 1. Among all methods in Table 1, Deeplabv3+ based on MobilenetV2 achieve best mIoU (38.1%), however, the latency is more than 1000 ms, which restrict its application on mobile devices.

Among these CNNs based baselines, the approach which takes mobilenetV3-large [16] as encoder and LR-ASPP as decoder, achieves good trade-off between computation (2.0 GFLOPs) and accuracy (33.1 mIoU). Following [16], we also reduce the channel counts of all feature layers in the last stage for further reducing the computation cost, denoted as MobileNetV3-Large-reduce. Based on the lighter backbone, LR-ASPP could achieve 32.3% mIoU with the lower latency (81 ms). Our small version of TopFormer is 3.8% more accurate compared to a LR-ASPP model with comparable latency. The tiny version of TopFormer could achieve comparable performance compared to LR-ASPP with $2\times$ less computation (0.6G vs. 1.3G). Lite-ASPP [4] is the reduced channel version of Deeplabv3+.

Among these ViT based baselines, HR-NAS-B [10] uses search techniques to introduce Transformer block into the HRNet [37] design, also achieves good trade-off between computation amount (2.2 GFLOPs) and accuracy (34.9 mIoU). Our small version of TopFormer is 1.2% more accurate compared to HR-NAS-B model with fewer computation. SegFormer [46] achieves great performance (37.4 mIoU) with fewer parameters (3.8M), although SegFormer adopts the efficient multi-head self-attention, the computation is still heavy due to a large number of tokens. Our base version of TopFormer could achieve comparable performance compared to SegFormer with more than $4\times$ less computation (1.8 GFLOPs vs. 8.4 GFLOPs).

To achieve real-time segmentation on the ARM-based mobile device, we resize the input image to 448×448 and feed it into TopFormer-tiny, the inference time is reduced to 32ms with a slight performance drop. To the best of our knowledge, it is the first ViT based method could achieve real-time segmentation on the ARM-based mobile device with competitive results.

Model	Params(M)	FLOPs(M)	mIoU
Baseline	0.3	573	22.5
+SASE	1.4	643	32.8
+FFNs	1.4	642	30.4
+PSP	1.7	654	27.4
+ASPP	2.3	698	28.2

Table 4 – Ablation studies on Scale-aware Semantics Extractor.

4.1.4 Ablation Study

We first conduct ablation experiments to discuss the influence of different components, including the token pyramid, Semantic Injection Module and Segmentation Head. Without loss of generality, all results are obtained by training on the training set and evaluation on the validation (val) set.

The influence of the Token Pyramid. Here, we discuss the Token Pyramid from two aspects, the influence of taking Token Pyramid as input and the influence of choosing Tokens from different scales as output. As reported in Table 2, we conduct the experiments that take stacked tokens from different scales as input of the Semantics Extractor, and take the last token as input of the Semantics Extractor, respectively. For fair comparison, we append a 1×1 convolution layer to expand the channels as same as the stacked tokens. The experimental results demonstrate the effectiveness of using the token pyramid as input.

After obtaining scale-aware semantics, the SIM will inject the semantics into the local Tokens. To pursuit better trade-off between the accuracy and the computation cost, we try to choose tokens from different scales for injecting. As shown in Table 3, using tokens from $\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}\}$ could achieve the best performance with the heaviest computation. Using tokens from $\{\frac{1}{16}, \frac{1}{32}\}$ achieves worse performance with the lightest computation. To achieve a good trade-off between the accuracy and the computation cost,

SigmoidAttn	SemInfo	FLOPs(G)	mIoU
✓		1.809	37.3
	✓	1.809	37.0
✓	✓	1.820	37.8

Table 5 – Ablation studies on Semantic Injection Module.

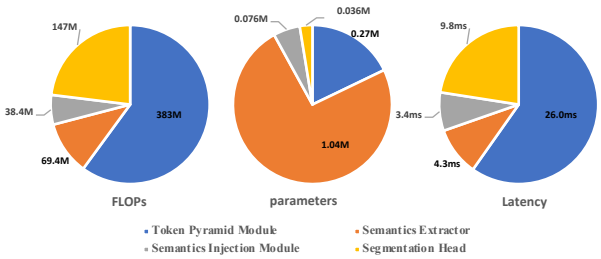

Figure 3 – The statistics of the computation, parameters, and latency. Latency and FLOPs are measured with input resolution 512×512.

we choose to use the tokens from $\{\frac{1}{8}, \frac{1}{16}, \frac{1}{32}\}$ in all other experiments.

The influence of the Scale-aware Semantics Extractor. Here, we have conducted experiments on Topformer-T to check the SASE. The results are shown in the Table 4. Here, we use Topformer without SASE as the baseline. Adding SASE brings about 10% mIoU gains, which is a significant improvement. To verify the multi-head self-attention module (MHSA) in the Transformer block, we remove all MHSA modules and add more FFNs for a fair comparison. The results demonstrate the MHSA could bring about 2.4% mIoU gains, which is an efficient and effective module under the careful architecture design. Meanwhile, we compare the SASE with the popular contextual models, such as ASPP and PPM, on the top of TPM. As shown in Table 4, "+SASE" could achieve better performance with much less computation cost than "+PSP" and "+ASPP". The experimental results demonstrate that the SASE is more appropriate for use in mobile devices.

The influence of the Semantic Injection Module and Segmentation Head. Due to the close relationship of Semantic Injection Module and Segmentation Head, we discuss these two together. Here, we discuss the design of Semantic Injection Module at first. As shown in Table 5, multiplying the local tokens and the semantics after a Sigmoid layer, denoted as "SigmoidAttn". Adding the semantics from Semantics Extractor into the corresponding local tokens, denoted as "SemInfo". Compared with "SigmoidAttn" and "SemInfo", adding "SigmoidAttn" and "SemInfo" simultaneously could bring pretty improvement with a little extra computation.

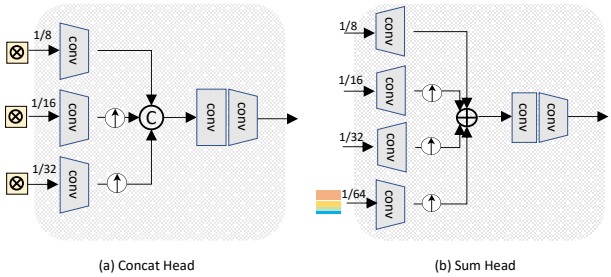

Figure 4 – Different segmentation heads in Token Pyramid Transformer. The Batch-norm layer and activation layer are omitted in the Figure.

Setting	Params(M)	mIoU
Concat Head	5.044	37.1
Sum Head	4.978	37.0
Ours	5.063	37.8

Table 6 – Ablation studies on Segmentation Head.

Here, we discuss the design of Segmentation Head. After passing the feature into Semantic Injection Module, the output hierarchical features are with both strong semantics and rich spatial details. The proposed segmentation head simply adds them together and then uses two 1×1 convolution layers to predict the segmentation map. We also design the other two Segmentation Heads, as shown in Figure 4. The "Sum Head" is identical to only adding "SemInfo" in SIM. The "Concat Head" uses a 1×1 convolution layer to reduce the channels of the outputs of SIM, then the features are concatenated together. Compared with "Concat Head" and "Sum Head", the current segmentation head could achieve better performance.

The influence of the width in SIM. In the paper, we donate the number of channels in SIM as M. Here, we study the influence of different M in SIM and find a suitable M to achieve a good trade-off. As shown in Table 7, the $M = 256, 192, 128$ achieve similar performance with very close computation. Thus, we set $M = 128, 192, 256$ in tiny, small and base model, respectively.

The influence of output stride. The tokens from different stages are pooled to fixed resolution, namely the output stride. The results of different resolutions are shown in Table 8. s32, s64, s128 are denoted that the resolution of pooled tokens are $\frac{1}{32\times32}, \frac{1}{64\times64}, \frac{1}{128\times128}$ of input size. Considering the trade-off of computation and accuracy, we choose s64 as the output stride of input tokens of the Semantics Extractor.

The statistics of the computation, parameters and latency. Here, we make the statistics of the computation,

M	mIoU	FLOPs(G)	Params(M)
128	37.3	1.5	4.9
192	37.4	1.6	5.0
256	37.8	1.8	5.1

Table 7 – Ablation studies on different #channels in SIM.

Stride	mIoU	FLOPs(G)	Latency(ms)
s32	38.8	2.6	159
s64	37.8	1.8	110
s128	35.7	1.6	100

Table 8 – Ablation studies on output strides.

Methods	Backbone	FLOPs(G)		$mIoU^{60}$	$mIoU^{59}$
		Backbone	Head		
DeepLabV3+	MBV2-s16	2.46	19.78	38.59	42.34
DeepLabV3+	ENet-s16	3.22	19.78	39.19	43.07
LR-ASPP	MBV3-s16	1.67	0.37	35.05	38.02
Ours	TopFormer-T	0.44	0.09	36.41	40.39
Ours	TopFormer-S	0.80	0.18	39.06	43.68
Ours	TopFormer-B	1.25	0.29	41.01	45.28

Table 9 – Results on Pascal Context *test* set.

parameters and latency of the proposed TopFormer-Tiny. As shown in Figure 3, although the Semantics Extractor has most parameters (74%), the FLOPs and actual latency of the Semantics Extractor is relatively low (about 10%).

4.1.5 Experiments on Pascal Context

We compare our TopFormer with the previous approaches on the Pascal Context test set in Table 9. We evaluate the performance over 59 categories and 60 categories (including background), respectively. It is obvious that our approach achieves better performance than all previous approaches based on CNNs or ViT with fewer computation. For better understanding, the FLOPs of the backbone and the head are measured, respectively. The proposed method can achieve best performance with the lightest backbone and head.

4.1.6 Experiments on COCO-Stuff

We compare our TopFormer with the previous approaches on the COCO-Stuff validation set in Table 10. The FLOPs of the backbone and the head are measured, respectively. It can be seen that our approach achieves the best performance, and the base version of TopFormer is 8% more accurate compared to a MobileNetV3 model with comparable computation.

4.2. Object Detection

To further demonstrate the generalization ability of the proposed TopFormer, we conduct object detection task on

Methods	Backbone	FLOPs(G)		mIoU
		Backbone	Head	
PSPNet	MobileNetV2-s8	7.84	45.10	30.14
DeepLabV3+	MobileNetV2-s16	2.46	23.44	29.88
DeepLabV3+	EfficientNet-s16	3.66	23.44	31.45
LR-ASPP	MobileNetV3-s16	1.89	0.48	25.16
Ours	TopFormer-T	0.49	0.15	28.34
Ours	TopFormer-S	0.89	0.29	30.83
Ours	TopFormer-B	1.38	0.45	33.43

Table 10 – Results on COCO-Stuff *test* set.

Backbone	mAP	FLOPs(G)		Params(M)	
		Backbone	Overall	Backbone	Overall
ShuffleNetv2	25.9	2.6	161	0.8	10.4
TopFormer-T	27.1	2.1	160	1.0	10.5
MobileNetV3	27.2	4.7	162	2.8	12.3
TopFormer-S	30.4	3.7	162	2.9	12.3
TopFormer-B	31.6	5.5	163	4.8	14.0

Table 11 – COCO object detecion based on RetinaNet.

COCO dataset. COCO consists of 118K images for training, 5K for validation and 20K for testing. We train all models on train2017 split and evaluate all methods on val2017 set. We choose RetinaNet [26] as object detection methods and adopt different backbones to produce feature pyramid. Our implementation is built on MMdetection [3] and Pytorch. For the proposed TopFormer, we replace the segmentation head with the detection head in RetinaNet. As shown in Table 11, The RetinaNet based on TopFormer could achieve better performance than MobileNetV3 and ShuffleNet with lower computation.

5. Conclusion and Limitations

In this paper, we present a new architecture for mobile vision tasks. With a combination of the advantages of CNNs and ViT, the proposed TopFormer achieves a good trade-off between accuracy and the computational cost. The tiny version of TopFormer could yield real-time inference on an ARM-based mobile device with competitive result. The experimental results demonstrate the effectiveness of the proposed method. The major limitation of TopFormer is the minor improvements on object detection. We will continue to promote the performance of object detection. Besides, we will explore the application of TopFormer in dense prediction in the future work.

Acknowledgement

This work was in part supported by NSFC (No. 61733007, No. 61876212, No. 62071127 and No. 61773176) and the Zhejiang Laboratory Grant (No. 2019NB0AB02 and No. 2021KH0AB05).

References

[1] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Cocostuff: Thing and stuff classes in context. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 2, 5

[2] Hanting Chen, Yunhe Wang, Chunjing Xu, Boxin Shi, Chao Xu, Qi Tian, and Chang Xu. Addernet: Do we really need multiplications in deep learning? In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1468–1477, 2020. 3

[3] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 8

[4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Eur. Conf. Comput. Vis.*, 2018. 5, 6

[5] Wuyang Chen, Xinyu Gong, Xianming Liu, Qian Zhang, Yuan Li, and Zhangyang Wang. Fasterseg: Searching for faster real-time semantic segmentation. *arXiv preprint arXiv:1912.10917*, 2019. 3

[6] Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Xiaoyi Dong, Lu Yuan, and Zicheng Liu. Mobile-former: Bridging mobilenet and transformer. *arXiv preprint arXiv:2108.05895*, 2021. 1

[7] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *arXiv:2104.13840*, 2021. 1

[8] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 5

[9] TNN Contributors. TNN: A high-performance, lightweight neural network inference framework. https://github.com/Tencent/TNN, 2019. 5

[10] Mingyu Ding, Xiaochen Lian, Linjie Yang, Peng Wang, Xiaojie Jin, Zhiwu Lu, and Ping Luo. Hr-nas: Searching efficient high-resolution neural architectures with lightweight transformers. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 3, 5, 6

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 2

[12] Ben Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. *Int. Conf. Comput. Vis.*, 2021. 1, 2, 4

[13] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 3

[14] Kai Han, Yunhe Wang, Qiulin Zhang, Wei Zhang, Chunjing Xu, and Tong Zhang. Model rubik's cube: Twisting resolution, depth and width for tinynets. *Adv. Neural Inform. Process. Syst.*, 33, 2020. 3

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 770–778, 2016. 1, 6

[16] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Int. Conf. Comput. Vis.*, pages 1314–1324, 2019. 2, 3, 5, 6

[17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2, 3, 4

[18] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Int. Conf. Comput. Vis.*, 2019. 2

[19] Zilong Huang, Youcheng Ben, Guozhong Luo, Pei Cheng, Gang Yu, and Bin Fu. Shuffle transformer: Rethinking spatial shuffle for vision transformer. *arXiv preprint arXiv:2106.03650*, 2021. 1, 4

[20] Zilong Huang, Yunchao Wei, Xinggang Wang, Humphrey Shi, Wenyu Liu, and Thomas Huang. Alignseg: Feature-aligned segmentation networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021. 3

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Int. Conf. Mach. Learn.*, 2015. 5

[22] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 5, 6

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inform. Process. Syst.*, 25:1097–1105, 2012. 1

[24] Hanchao Li, Pengfei Xiong, Haoqiang Fan, and Jian Sun. Dfanet: Deep feature aggregation for real-time semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 9522–9531, 2019. 3

[25] Xiangtai Li, Ansheng You, Zhen Zhu, Houlong Zhao, Maoke Yang, Kuiyuan Yang, Shaohua Tan, and Yunhai Tong. Semantic flow for fast and accurate scene parsing. In *Eur. Conf. Comput. Vis.*, pages 775–793, 2020. 3

[26] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Int. Conf. Comput. Vis.*, pages 2980–2988, 2017. 8

[27] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Eur. Conf. Comput. Vis.* Springer, 2014. 2

[28] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *Int. Conf. Comput. Vis.*, 2021. 1, 2

[29] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2015. 5

[30] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Eur. Conf. Comput. Vis.*, 2018. 3, 6

[31] Sachin Mehta and Mohammad Rastegari. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint arXiv:2110.02178*, 2021. 1, 2

[32] Sachin Mehta, Mohammad Rastegari, Linda Shapiro, and Hannaneh Hajishirzi. Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 9190–9200, 2019. 3

[33] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2014. 2, 5

[34] Vladimir Nekrasov, Hao Chen, Chunhua Shen, and Ian Reid. Fast neural architecture search of compact semantic segmentation models via auxiliary cells. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 9126–9135, 2019. 3

[35] Marin Orsic, Ivan Kreso, Petra Bevandic, and Sinisa Segvic. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12607–12616, 2019. 3

[36] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4510–4520, 2018. 2, 3, 4, 5, 6

[37] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 6

[38] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Int. Conf. Mach. Learn.*, pages 6105–6114, 2019. 3

[39] Mingxing Tan and Quoc V Le. Efficientnetv2: Smaller models and faster training. *arXiv:2104.00298*, 2021. 3

[40] Yehui Tang, Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chao Xu, and Dacheng Tao. Patch slimming for efficient vision transformers. *arXiv:2106.02852*, 2021. 1

[41] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv:2012.12877*, 2020. 2

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Adv. Neural Inform. Process. Syst.*, 2017. 1

[43] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *Int. Conf. Comput. Vis.*, 2021. 1

[44] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 2

[45] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021. 1

[46] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Adv. Neural Inform. Process. Syst.*, 2021. 5, 6

[47] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. *arXiv:1808.00897*, 2018. 3

[48] Kun Yuan, Shaopeng Guo, Ziwei Liu, Aojun Zhou, Fengwei Yu, and Wei Wu. Incorporating convolution designs into visual transformers. *arXiv preprint arXiv:2103.11816*, 2021. 4

[49] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis E.H. Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Int. Conf. Comput. Vis.*, 2021. 2

[50] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. 2020. 5

[51] Yuhui Yuan, Rao Fu, Lang Huang, Weihong Lin, Chao Zhang, Xilin Chen, and Jingdong Wang. Hrformer: High-resolution transformer for dense prediction. *arXiv preprint arXiv:2110.09408*, 2021. 1

[52] Bowen Zhang, Yifan Liu, Zhi Tian, and Chunhua Shen. Dynamic neural representational decoders for high-resolution semantic segmentation. *arXiv:2107.14428*, 2021. 3

[53] Ting Zhang, Guo-Jun Qi, Bin Xiao, and Jingdong Wang. Interleaved group convolutions. In *Int. Conf. Comput. Vis.*, pages 4373–4382, 2017. 3

[54] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6848–6856, 2018. 3

[55] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.* 2

[56] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. In *Eur. Conf. Comput. Vis.*, pages 405–420, 2018. 3

[57] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017. 5

[58] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. *arXiv preprint arXiv:2012.15840*, 2020. 1

[59] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through

ade20k dataset. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017. 2, 5

[60] Daquan Zhou, Qibin Hou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. Rethinking bottleneck structure for efficient mobile network design. In *Eur. Conf. Comput. Vis.* Springer, 2020. 3